JOURNAL OF COMPUTER AND SYSTEM SCIENCES 35, 72–95 (1987)

Union and Actualization of Module Specifications: Some Compatibility Results

FRANCESCO PARISI-PRESICCE

Department of Mathematics, University of Southern California, Los Angeles, California 90089-1113

Received July 15, 1985; revised June 12, 1986

In recent papers, a notion of algebraic module specification has been introduced, along with operations to combine them. Here we take a closer look at the operations of union of module specifications with import and export interfaces and of actualization of the parameter specification that the two interfaces share. We show that both the standard and parametrized actualization are compatible with the previously introduced notion of submodule and with the union operation. © 1987 Academic Press, Inc.

INTRODUCTION

In the framework of modular development of large software systems [21, 27], a new algebraic specification concept called a "module," was introduced by Ehrig and Weber [11] and developed further in [1, 2, 6]. The module concept extends the notion of the abstract data type as formulated in [19] and combines the notions of parameterization as in [8, 14], of implementation as in [7, 4], and of information hiding in the sense of [17]. The first step toward the formalization of modules with interfaces can be found in [16], where hidden functions are treated by adding to the data type an export interface containing only the visible operations; this idea has been carried one step further by isolating an import interface to produce the modules discussed here. Other algebraic approaches to the problem of information hiding include the notions of behavioral abstraction and observability as in [15, 22, 24, 25].

An abstract module is an abstract data type [19] equipped with an import interface and an export interface. The import interface is given by an algebraic specification with loose semantics and represents the operations available inside the module (these operations have either been previously specified or been provided by the system as in the package STANDARD of the Ada language [26]). The export interface, also given by a specification with loose semantics, consists of the sorts and operations available to the user of the module (in Ada, this is the *visible part* of the package, i.e., the first list of declarative items in the package specification). The two interfaces are combined in the body of the module, which is intended to provide an implementation of the sorts and operations of the export interface using

0022-0000/87 $3.00

those of the import interface. The operations in the body which are not in the export interface are considered "hidden" and not accessible from outside the module. The body is also represented by a specification, whose semantics is taken to be a free construction on import-algebras. Finally, the two interfaces are allowed to share a common parameter part, not modified by the implementation in the body. The semantics of a module specification is a functor from the category of import-algebras to the category of export-algebras. As a functor, it transforms isomorphic import-algebras into isomorphic export-algebras.

The three basic operations on modules are composition, union, and actualization. In the composition M1 · M2 of two module specifications, the export and parameter of M2 are "matched" with the export and parameter of M1 via a pair of specification morphisms. The new module has the import interface of M2, the export interface and parameter part of M1, and a body which is a "union" of the bodies of M1 and M2. We will not consider this operation here and refer the reader to [1, 5, 11] for a precise definition of the composition and the results on its induced semantics.

The operation of actualization consists of replacing the parameter part of a module specification with an "actual" parameter using, again, a specification morphism. The import (export) interface of the new module is obtained by combining the old import (export) with the actual parameter and the body by combining the new import, or, equivalently, the new export (see Section 3) with the old body. The new interfaces share no parameter or a new parameter part, depending on whether the actualization is standard or parametrized.

The union of two module specifications M1 and M2 is a module specification whose interfaces, parameter part and body part, are the "union" of the corresponding specifications of the two modules. The union of disjoint modules or the "disjoint" union of modules that share a common part, which we are willing to duplicate, poses no difficulties. But special care is needed if the module specifications share a common part (possibly "translated" using a specification morphism) of the interfaces and/or of the parameter part which is not to be duplicated. The union, in different cases, has been introduced in [2].

In this paper, we take a closer look at the operations of union and actualization of modules and show that the two operations are compatible. In Sections 1 and 2, we review the definition and basic results of module and submodule specification and the operation of the union of modules. The interaction of actualization with submodule specifications and the union operation is discussed in Sections 3 and 4, respectively. In Section 5, we consider a simple case of parametrized actualization and its behavior with respect to the union of modules. The discussions in this paper focus primarily on the basic algebraic case.

1. Modules and Submodules

In this section, we review the basic notions of module specification with import and export interfaces as introduced by Ehrig and Weber in [11] and of submodule

specification as presented first in [2]. For a full treatment, see [1]. All the algebraic specifications considered in this paper are of the form (S, Σ, E), where S is a set of sorts, Σ is a family of sets of operator symbols indexed by $S^* \times S$, and E is a set of equations (or universal Horn clauses). If SPEC is a specification, we denote by **SPEC** the category whose objects are SPEC-algebras and whose morphisms are SPEC-homomorphisms. Specification morphisms f: SPEC $\to$ SPEC′ are pairs (f_S, f_Σ) of functions preserving the equations in E (see [9] for details). The category of specifications and specification morphisms is denoted by CATSPEC and it is known [4, 9] to be closed under pushouts.

1.1. DEFINITION. A module specification MOD is a 4-tuple (PAR, IMP, BOD, EXP) of specifications representing the parameter part, the import interface, the body, and the export interface, respectively, along with specification morphisms i, s, v, and e making the following diagram commutative

$$\begin{array}{ccc} \text{PAR} & \xrightarrow{e} & \text{EXP} \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle v} \\ \text{IMP} & \xrightarrow[s]{} & \text{BOD} \end{array} \quad . \tag{1}$$

The morphisms e and s are assumed to be injective. Each of the specifications of MOD determines a category of algebras and each specification morphism a forgetful functor between the appropriate categories. The diagram above then gives rise to the following commutative diagram in the category of SPEC-algebras categories and forgetful functors.

$$\begin{array}{ccc} \mathbf{PAR} & \xleftarrow{V_e} & \mathbf{EXP} \\ {\scriptstyle V_i}\uparrow & & \uparrow{\scriptstyle V_v} \\ \mathbf{IMP} & \xleftarrow[V_s]{} & \mathbf{BOD} \end{array} \quad . \tag{2}$$

Moving away from specifications (more precisely, disregarding the E part of the specifications) we can think of a module as a 4-tuple of classes of algebras (**P**, **I**, **B**, **E**) and four functors between these classes making a diagram like diagram (2) commute. If the parameter part is empty, this corresponds to the notion of a "package" in Ada [26]. The visible part, declared in the package specification, defines the sorts and operations (no equations) of the export interface, giving rise to the class of algebras **E**. The operations and sorts used (but not defined) in the body of the package constitute the import interface which is the signature of the class **I**. The body contains the signatures of both interfaces along with auxiliary sorts and operations defining the signature of the class **B**. The user of such a module will have available a *specific* algebra $E \in \mathbf{E}$ corresponding to an "instance" of the package, i.e.,

a specific triple (I, B, E) of algebras, belonging to the appropriate classes, determined by the particular implementation of the package STANDARD and the particular definition of the operations and sorts of the import interface. A similar view can be taken of the "generics" of Ada, where the parameter part is nonempty. An example, illustrating the similarities, can be found in [1].

Returning to our specifications and the definition of MOD, we now define the semantics of a module specification.

1.2. DEFINITION. Let MOD be a module specification as in Definition 1.1, with the commutative diagrams (1) and (2) and let FREE: **IMP** → **BOD** denote the free functor associated with the forgetful functor V_s (i.e., its left adjoint). The (unrestricted) *semantics* of MOD is the functor

$$\mathrm{SEM} = V_b \cdot \mathrm{FREE}: \mathbf{IMP} \to \mathbf{EXP}.$$

The *restricted semantics* of MOD is the functor

$$\mathrm{RSEM} = R_e \cdot \mathrm{SEM}: \mathbf{IMP} \to \mathbf{EXP},$$

where R_e: **EXP** → **EXP** is the restriction functor defined for every $E \in \mathbf{EXP}$ by

$$R_e(E) = \bigcap \{B \in \mathbf{EXP}: B \subset E,\ V_e(E) = V_e(B)\}.$$

A semantical constraint is imposed on the free functor FREE: we assume that FREE is *strongly persistent*, i.e., that $V_s(\mathrm{FREE}(A)) = A$ for any algebra A in **IMP**.

1.3. *Comments.* The constraint on the functor FREE can be interpreted as requiring that the evaluation of any function defined in the body and with a range sort $s_S(r)$, for r an import sort, must always terminate with a unique value when using the defining "equations" as rewrite rules. Persistency of FREE implies, by the commutativity of diagram (2), that the parameter part $V_i(A)$ of any import algebra A is the same as the parameter part $V_e(\mathrm{SEM}(A))$ of the corresponding export algebra $\mathrm{SEM}(A)$. The restriction functor R_e, when applied to $\mathrm{SEM}(A)$, reduces the carrier of the algebra available to the user of the module to those elements which can be constructed from the parameter part with the sole use of the visible operations (those in the export interface). By definition, the functor R_e does not modify $V_e(E)$ and so again we have $V_i(A) = V_e(\mathrm{RSEM}(A))$.

In [11, 1], the functor FREE is required to be strongly conservative (i.e., strongly persistent and preserve injective morphisms) when using the restricted semantics. Intuitively, it requires that an IMP-subalgebra B of an IMP-algebra A produce a subalgebra $\mathrm{FREE}(B)$ of $\mathrm{FREE}(A)$ and it implies, since forgetful functors preserve monomorphisms, that $(R)\mathrm{SEM}(B)$ is a subalgebra of $(R)\mathrm{SEM}(A)$. It is also a technical condition needed to prove that the restricted semantic of the com-

position M1 · M2 of module specifications is the composition of the respective restricted semantics. We will not discuss composition in detail (see [1]) and the requirement is not needed in treating union and actualization.

We now give a simple example of a module specification to illustrate the two definitions. Let MOD be given by

PAR = **sort**: elem

IMP = PAR +
sort: list
opns: EMPTY: → list
ADD: elem list → list

EXP = PAR +
sort: list
opns: EMPTY: → list
MAKE: elem → list
REVERSE: list → list
JOIN: list list → list
eqns: REVERSE (EMPTY) = EMPTY
REVERSE (MAKE(x)) = MAKE(x)
REVERSE (REVERSE(x)) = x
JOIN (EMPTY, x) = x

BOD = IMP ∪ EXP +
eqns: MAKE(e) = ADD(e, EMPTY)
JOIN (x, EMPTY) = x
JOIN (ADD(e, x), y) = ADD(e, JOIN(x, y))
REVERSE(ADD(e, x)) = JOIN (REVERSE(x), MAKE(e))

with inclusions for the specification morphisms in diagram (1). Notice that PAR is not the largest specification shared by IMP and EXP, i.e., diagram (1) is not required to be a pullback but only to be commutative.

The two semantics of MOD transform IMP-algebras into EXP-algebras. If X is any set and A is the IMP-algebra with

$$A_{\text{elem}} = X, \qquad A_{\text{list}} = \text{power set of } X$$

$$\text{EMPTY}_A = \varnothing \qquad \text{and} \qquad \text{ADD}_A(x, P) = \{x\} \cup P$$

then $B = \text{SEM}(A)$ is the EXP-algebra with

$$B_{\text{elem}} = X, \qquad B_{\text{list}} = \text{power set of } X$$

$$\text{EMPTY}_B = \varnothing, \qquad \text{MAKE}_B(x) = \{x\}$$

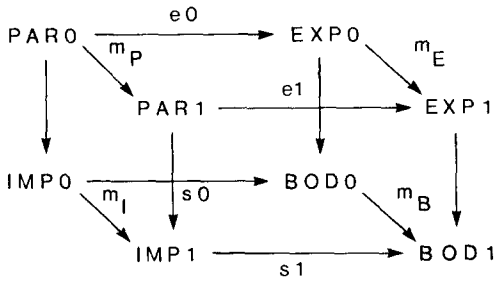
$$\text{REVERSE}_B(P) = P \qquad \text{and} \qquad \text{JOIN}_B(P, Q) = P \cup Q.$$

The restricted semantics RSEM provides a different EXP-algebra $C = \text{RSEM}(A) = R_e(B)$. While the parameter part C_{elem} is the same as A_{elem} and the

operations of C are those of B, the carrier C_{list} is reduced to the collection of finite subsets of X, since these are the only elements of B_{list} representable using only X and the operations EMPTY_C, MAKE_C, REVERSE_C, and JOIN_C.

The notion of submodule is needed to define the union of module specifications. From an intuitive point of view, a module MOD0 is a submodule of a module MOD1 if, given a reduction of an import algebra of MOD1, it produces an export algebra which is a reduct of the corresponding export algebra of MOD1 and this is done by using a (translated) part of the body of MOD1. We now make this formal.

1.4. DEFINITION. A module specification MOD0 = (PAR0, IMP0, BOD0, EXP0) is a *submodule specification* of MOD1 = (PAR1, IMP1, BOD1, EXP1) if there exist four injective specification morphisms m_P: PAR0 → PAR1; m_I: IMP0 → IMP1; m_B: BOD0 → BOD1; and m_E: EXP0 → EXP1 such that in the diagram



the top, the bottom, and the two side squares commute (the front and back squares commute by definition of module specification).

Assumptions. We have already assumed (in 1.2) that both free functors, FREE0 and FREE1, are strongly persistent. For MOD0 to be considered a submodule of MOD1, the free construction FREE0 should reflect the construction FREE1 on the restricted import interface; we therefore require that $V_{m_B} \cdot \text{FREE1} = \text{FREE0} \cdot V_{m_I}$. This is sufficient to relate the (unrestricted) semantics of MOD0 and MOD1, while when using the restricted semantics we will also add the requirement that $V_{m_E} \cdot R_{e1} = R_{e0} \cdot V_{m_E}$.

1.5. THEOREM. *Given a submodule specification MOD0 of a module specification MOD1 satisfying the assumptions above, we have:*

(i) $V_{m_E} \cdot \text{SEM1} = \text{SEM0} \cdot V_{m_I}$ *and*

(ii) $V_{m_E} \cdot \text{RSEM1} = \text{RSEM0} \cdot V_{m_I}$.

The proof of this theorem is straightforward using the definition of SEM and RSEM, the commutativity of the forgetful functors induced by the diagram in Definition 1.4, and the above assumptions. As already mentioned in [2], the notion of submodule specification is closely related to that of "refinement" and of

"extension" of module specifications introduced in [11]. Finally, the assumption that $V_{m_E} \cdot R_{el} = R_{e0} \cdot V_{m_E}$ can be thought of as requiring the specification EXP1 to be an "extension" (translated by m_E) of the specification EXP0 w.r.t. m_P in the sense of [7, 8].

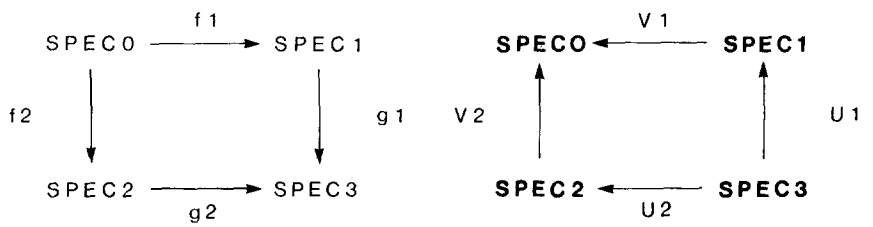
2. Union of Modules and Amalgamated Sums

As mentioned in the Introduction, there are three basic operations that can be performed on module specifications: composition (discussed in [11] and [1]), union (introduced in [2] and reviewed in this section), and actualization (presented in [11] and outlined in the next two sections).

Given two modules MOD1 and MOD2, their union provides a new module whose interfaces are obtained by combining the corresponding interfaces of MOD1 and MOD2. If the two modules are disjoint or if the two modules share a common part which is expected to be duplicated (using, for example, a "renaming" declaration as in Ada), then the composite module MOD3 can be formed in a straightforward manner by taking the disjoint union of the corresponding specifications MOD3 = (PAR1 + PAR2, IMP1 + IMP2, BOD1 + BOD2, EXP1 + EXP2). It is easy to see that the assumptions made in 1.2 are again satisfied. Any import-algebra of MOD3 is the disjoint union of an IMP1-algebra and an IMP2-algebra and the semantics of the resulting module MOD3 is given by SEM3(A1 + A2) = SEM1(A1) + SEM2(A2), and similarly for the restriction semantics.

If the two module specifications share a common part which is not intended to be duplicated, such as a common parameter part to be instantiated with the same actual parameter, or a common part of the interfaces such as the boolean type, then a more careful definition of union must be developed. The semantics of the union of such module specifications requires the notion of an *amalgamated sum* of algebras, which we define next.

Let the diagrams



be a pushout diagram in the category of specifications and specification morphisms and its corresponding pullback diagram in the category of SPEC-algebra categories and forgetful functors.

2.1. DEFINITION. If $Ai \in \mathbf{SPECi}$ for $i = 0, 1, 2$ are algebras satisfying $V1(A1) = A0 = V2(A2)$ then the *amalgamated sum* $A1 +_{A0} A2$ of $A1$ and $A2$ w.r.t. $A0$ is the unique SPEC3-algebra $A3$ defined by

$$(A3)_s = \begin{cases} A1_{s1} & \text{if } g1(s1) = s \\ A2_{s2} & \text{if } g2(s2) = s \end{cases}$$

$$\sigma_{A3} = \begin{cases} \sigma 1_{A1} & \text{if } g1(\sigma 1) = \sigma \\ \sigma 2_{A2} & \text{if } g2(\sigma 2) = \sigma \end{cases}$$

for all sorts $s \in S3$ and operators $\sigma \in \Sigma 3$.

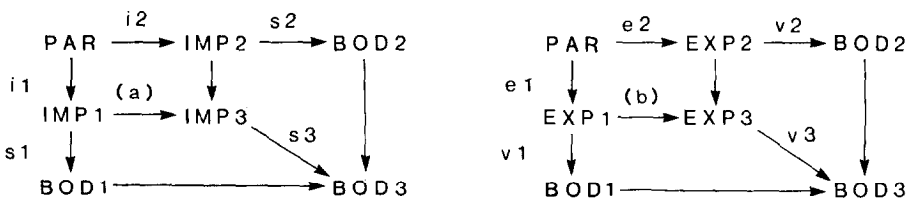
It has been argued in [2] and proved in [9] that $A3$ is a well-defined SPEC3-algebra. It has also been shown there that $A3$ can be defined implicitly using the pullback diagram above in the sense that $A3$ is the *unique* SPEC3-algebra such that $U1(A3) = A1$, $U2(A3) = A2$, and if $B \in \mathbf{SPEC}$ for a specification SPEC with forgetful functors Fi: **SPEC** $\rightarrow$ **SPECi**, satisfying $Fi(B) = Ai$ for $i = 1, 2$, then there exists a unique forgetful functor F: **SPEC** $\rightarrow$ **SPEC3** such that $Fi = Vi \cdot F$ and $F(B) = A3$. This implies that

$$\mathbf{SPEC3} = \{A1 +_{A0} A2 \colon Ai \in \mathbf{SPECi},\ V1(A1) = A0 = V2(A2)\}$$

and we will refer to this last class of algebras as **SPEC1** $+_{\mathbf{SPEC0}}$ **SPEC2**. It is also shown in [2] that, in the appropriate category **UAlg**, each amalgamated sum of two algebras can be viewed as a pushout. Note, finally, that if SPEC0 $= \varnothing = A0$, then $A1 +_{A0} A2$ is the disjoint union of $A1$ and $A2$.

We are now ready to given the precise definition of union of module specifications and its semantics when the modules share a common parameter part.

2.2. DEFINITION. The union of MODi = (PAR, IMPi, BODi, EXPi) for $i = 1, 2$, with respect to the shared parameter part PAR is the module specification MOD3 = (PAR, IMP3, BOD3, EXP3) denoted by MOD1 $+_{\text{PAR}}$ MOD2 and given by the pushout diagrams:



IMP3 and EXP3 are given by the pushout diagrams (a) and (b), respectively, while BOD3 is obtained as a pushout object of the outer square (they are the same by commutativity of the SPEC-diagrams of MOD1 and MOD2). The existence and uniqueness of $s3$ and $v3$ is guaranteed by the pushout properties of IMP3 and EXP3, respectively. It is easy to show that the diagram of MOD3 still commutes and satisfies the assumptions in 1.2.

It has been argued earlier that every import-algebra of MOD3 has the form $A1 +_P A2$ for $Ai \in \mathbf{IMPi}$ and $P \in \mathbf{PAR}$ and that every EXP3-algebra can be written as $E1 +_P E2$ for $Ei \in \mathbf{EXPi}$. This leads to the following theorem relating the semantics of MOD3 with those of MOD1 and MOD2.

2.3. THEOREM. *The semantics of* $\mathrm{MOD3} = \mathrm{MOD1} +_{\mathrm{PAR}} \mathrm{MOD2}$ *as in Definition* 2.2 *are given by*

(ii) $\mathrm{SEM3}(A1 +_P A2) = \mathrm{SEM1}(A1) +_P \mathrm{SEM2}(A2)$

(ii) $\mathrm{RSEM3}(A1 +_P A2) = \mathrm{RSEM1}(A1) +_P \mathrm{RSEM2}(A2)$.

We will abbreviate (i) *and* (ii) *by writing* $\mathrm{SEM3} = \mathrm{SEM1} +_{\mathrm{PAR}} \mathrm{SEM2}$ *and* $\mathrm{RSEM3} = \mathrm{RSEM1} +_{\mathrm{PAR}} \mathrm{RSEM2}$, *respectively.*

The case where the two module specification $\mathrm{MOD}i = (\mathrm{PAR}i, \mathrm{IMP}i, \mathrm{BOD}i, \mathrm{EXP}i)$ share a subparameter part PAR0 with specification morphisms pi: $\mathrm{PAR0} \to \mathrm{PAR}i$, for $i = 1, 2$, can be handled in a similar way by replacing, for example, in the diagram (a),

$$i2\colon \mathrm{PAR} \to \mathrm{IMP2} \text{ with } i2 \cdot p2\colon \mathrm{PAR0} \to \mathrm{PAR2} \to \mathrm{IMP2}$$

and

$$i1\colon \mathrm{PAR} \to \mathrm{IMP1} \text{ with } i1 \cdot p1\colon \mathrm{PAR0} \to \mathrm{PAR1} \to \mathrm{IMP1}$$

and forming again the pushouts. It can be shown again that $\mathrm{SEM3} = \mathrm{SEM1} +_{\mathrm{PAR0}} \mathrm{SEM2}$ and similarly for RSEM3. The most general case to be considered is the union of two modules which share a common submodule that should not be duplicated.

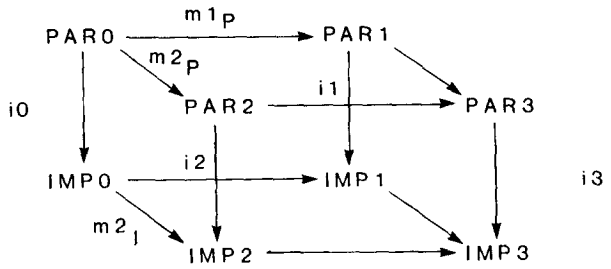
2.4. DEFINITION. Let $\mathrm{MOD0} = (\mathrm{PAR0}, \mathrm{IMP0}, \mathrm{BOD0}, \mathrm{EXP0})$ be a submodule specification of $\mathrm{MOD}j = (\mathrm{PAR}j, \mathrm{IMP}j, \mathrm{BOD}j, \mathrm{EXP}j)$ for $j = 1, 2$ with specification morphisms

$$mj_P\colon \mathrm{PAR0} \to \mathrm{PAR}j, \qquad mj_I\colon \mathrm{IMP0} \to \mathrm{IMP}j,$$

$$mj_B\colon \mathrm{BOD0} \to \mathrm{BOD}j \quad \text{and} \quad mj_E\colon \mathrm{EXP0} \to \mathrm{EXP}j$$

satisfying the commutative properties of Definition 1.4. The *union of MOD*1 *and MOD*2 *w.r.t. MOD*0, denoted by $\mathrm{MOD1} +_{\mathrm{MOD0}} \mathrm{MOD2}$, is the module specification $\mathrm{MOD3} = (\mathrm{PAR3}, \mathrm{IMP3}, \mathrm{BOD3}, \mathrm{EXP3})$, where each of its specifications is obtained by forming the pushout of the corresponding

specifications in MOD0, MOD1, and MOD2 with the appropriate specification morphisms. The specifications PAR3 and IMP3, for example, are given by the diagram



In this diagram, the back and left side squares are commutative since MOD0 is a submodule specification of MOD1 and MOD2; the top and bottom squares are pushout diagrams defining PAR3 and IMP3, respectively. The specification morphism $i3$ is unique by the universal property of the top diagram and it makes the front and right side diagrams commutative. The diagrams giving BOD3 and EXP3 can be constructed in a similar manner. It has been shown [2] that the resulting module specification satisfies the syntactical and semantical conditions of Definitions 1.1 and 1.2.

If we take PAR0 = IMP0 = BOD0 = EXP0, then the above definition reduces to the union of module specifications sharing a common subparameter part. If the two module specification MOD1 and MOD2 share only part of the import (resp. export) interface, then we take PAR0 = EXP0 = $\varnothing$ (resp. PAR0 = IMP0 = $\varnothing$) and IMP0 = BOD0 (resp. EXP0 = BOD0).

2.5. THEOREM. *The semantics of the module specification* $\mathrm{MOD3} = \mathrm{MOD1} +_{\mathrm{MOD0}} \mathrm{MOD2}$ *are uniquely given by*

(i) $\mathrm{SEM3} = \mathrm{SEM1} +_{\mathrm{SEM0}} \mathrm{SEM2}$

(ii) $\mathrm{RSEM3} = \mathrm{RSEM1} +_{\mathrm{RSEM0}} \mathrm{RSEM2}$.

Stated informally, the (unrestricted) semantics of MOD3 is the amalgamated sum of the semantics of MOD1 and MOD2 w.r.t. the semantics of MOD0, where $\mathrm{SEM1} +_{\mathrm{SEM0}} \mathrm{SEM2}$: **IMP3** → **EXP3** is defined by $(\mathrm{SEM1} +_{\mathrm{SEM0}} \mathrm{SEM2})(A1 +_{A0} A2) = \mathrm{SEM1}(A1) +_{\mathrm{SEM0}(A0)} \mathrm{SEM2}(A2)$ for $Ai \in$ **IMPi**, $i = 0, 1, 2$ using the fact, stated earlier in this section, that $\mathbf{IMP3} = \mathbf{IMP1} +_{\mathbf{IMP0}} \mathbf{IMP2}$. The proof of both parts of this theorem makes explicit use of the assumptions made at the end of Definition 1.4 concerning submodule specifications.

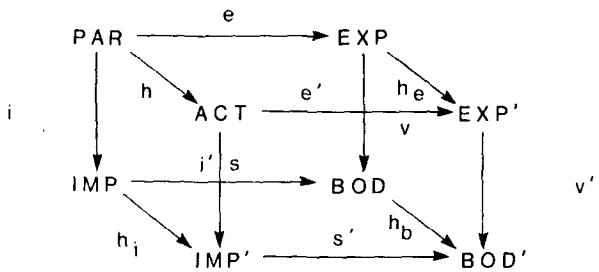
3. ACTUALIZATION AND SUBMODULES

In this section, we first review the operation of actualization of module specifications (as introduced in [11, 1]) and then show its compatibility with the

notion of submodule specification. We restrict our attention here to standard actualization, postponing the discussion on a simple case of parameterized actualization to a later section.

The operation of (standard) actualization of a module specification MOD = (PAR, IMP, BOD, EXP) consists of "replacing" the parameter part PAR by a non-parameterized specification ACT via a parameter passing morphism h: PAR $\to$ ACT. The result is a parameterless module specification whose interfaces are obtained by "gluing" the respective interfaces of MOD with ACT via the common PAR part. The sorts and operations of ACT are also added to the BOD part of the module specification. If in Definition 1.1 we take PAR = IMP and EXP = BOD, we obtain a parameterized specification [8, 4, 14]. In this case, actualization reduces to parameter passing in the sense of [7–9].

3.1. DEFINITION. The result of actualizing the parameter part of the module specification MOD = (PAR, IMP, BOD, EXP) by an actual parameter specification ACT with the parameter passing morphism h: PAR $\to$ ACT is denoted by act_h(ACT, MOD) and is the (parameterless) module specification MOD′ = ($\emptyset$, IMP′, BOD′, EXP′) given by the diagram

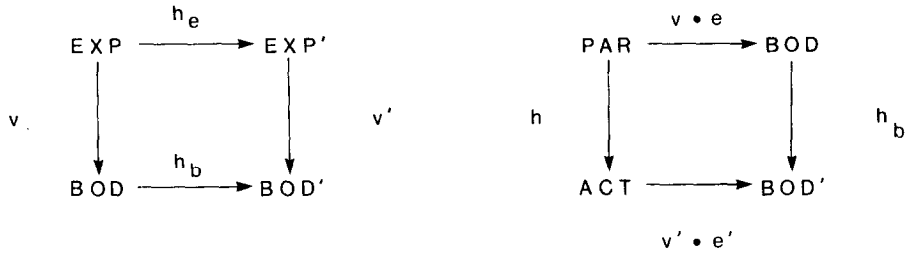


In this diagram, the top and left side squares are pushout diagrams defining EXP′ and IMP′, respectively, and BOD′ is obtained as the pushout object of the bottom square. The existence and uniqueness of the specification morphism v' is guaranteed by the pushout property of EXP′, and v' makes the front and right side squares commutative. The new free construction FREE′: **IMP′ $\to$ BOD′** is again strongly persistent by the Extension Lemma in [8, 9].

For notational convenience, we denote by e': ACT $\to$ EXP′ and i': ACT $\to$ IMP′ the specification morphisms induced by the pushout diagrams, even though they are *not* the specification morphisms $\emptyset \to$ EXP′ and $\emptyset \to$ IMP′ of the module specification MOD′.

In the construction of MOD′, BOD′ is obtained by "gluing" IMP′ and BOD and v' is an induced specification morphism. Not only the roles of IMP′ and EXP′ can be reversed, but BOD′ can be obtained directly as the gluing of ACT and BOD.

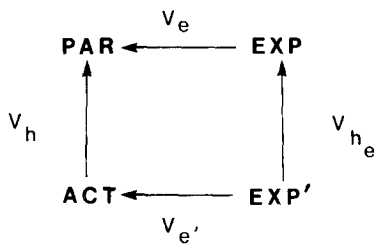
3.2. LEMMA. *The following are pushout diagrams*



The diagram in Definition 3.1 suggests a connection with the notion of submodule specification. We prove it next.

3.3. PROPOSITION. MOD = (PAR, IMP, BOD, EXP) *is a submodule specification of* MOD(ACT) = (ACT, IMP', BOD', EXP').

Proof. The specification morphisms h, h_i, h_b, and h_e play the role of m_P, m_I, m_B, and m_E of Definition 1.4. The constraint $V_{h_b} \cdot \text{FREE}' = \text{FREE} \cdot V_{h_i}$ is satisfied using the Extension Lemma in [9], since the bottom square in the diagram of Definition 3.1 is a pushout diagram. To show that $V_{h_e} \cdot R_{e'} = R_e \cdot V_{h_e}$, consider the pullback diagram induced by the pushout definition of EXP':



Then notice that, by definition of the restriction functor, $V_{e'} \cdot R_{e'} = V_{e'}$ and $V_e \cdot R_e = V_e$ and that both $V_{h_e} \cdot R_{e'}$ and $R_e \cdot V_{h_e}$ are functors from **EXP'** to **EXP** satisfying $V_e \cdot V_{h_e} \cdot R_{e'} = V_h \cdot V_{e'} = V_e \cdot R_e \cdot V_{h_e}$. By the pullback property of **EXP'**, there exists *only one* such functor and the proof is complete.

3.4. THEOREM. *The unrestricted semantics* $\text{act}_h(\text{ACT}, \text{MOD})$ *is uniquely given by*

$$\text{SEM}' = \text{id}_{\text{ACT}} + _{\text{id}_{\text{PAR}}} \text{SEM},$$

where SEM *is the semantics of* MOD, *and* id_{ACT} *and* id_{PAR} *are the identity functors on the categories* **ACT** *and* **PAR**.

Proof. By definition, each IMP'-algebra I' is of the form $A + _P I$ for $A \in$ **ACT**,

$P \in \mathbf{PAR}$, and $I \in \mathbf{IMP}$ with $V_h(A) = P = V_i(I)$. Since MOD(ACT) and act_h(ACT, MOD) have the same unrestricted semantics, by 3.3 and 1.5 we have

$$V_{h_e}(\mathrm{SEM}'(I)) = \mathrm{SEM}(V_{h_i}(I')) = \mathrm{SEM}(I)$$

and

$$V_{e'}(\mathrm{SEM}'(I')) = V_{e'} \cdot V_{v'}(\mathrm{FREE}'(I')) = V_{i'}V_{s'}(\mathrm{FREE}'(I')) = V_{i'}(I') = A.$$

But we also have $V_{h_e}(A +_P \mathrm{SEM}(I)) = \mathrm{SEM}(I)$ and $V_{e'}(A +_P \mathrm{SEM}(I)) = A$. By the uniqueness of the amalgamated sum, $\mathrm{SEM}(I') = A +_P \mathrm{SEM}(I)$.

The correspondence between the restricted semantics of MOD and act_h(ACT, MOD) is not as direct, if the semantics of ACT is not taken into account. We will come back to this point in Section 5. For now we can only establish the following result, which also appears in [11] with a direct proof.

3.5. THEOREM. *The restricted semantics* RSEM′ *of* act_h(ACT, MOD) *satisfies*

$$V_{h_e} \cdot \mathrm{RSEM}' = \mathrm{RSEM} \cdot V_{h_i} \qquad \textit{and} \qquad V_{e'} \cdot \mathrm{RSEM}' = V_{i'}$$

when the domain of the functors is restricted to the algebras $A \in \mathbf{IMP}'$, *where* $V_{i'}(A)$ *is a homomorphic image of* T_{ACT}, *the initial algebra of* **ACT**.

Proof. For such an algebra $A \in \mathbf{IMP}'$, $R'(\mathrm{SEM}'(A)) = R_{e'}(\mathrm{SEM}'(A))$, where R' is the restriction functor of $\varnothing \to \mathrm{EXP}'$. The conclusion now follows from Theorem 1.5 and Proposition 3.3.

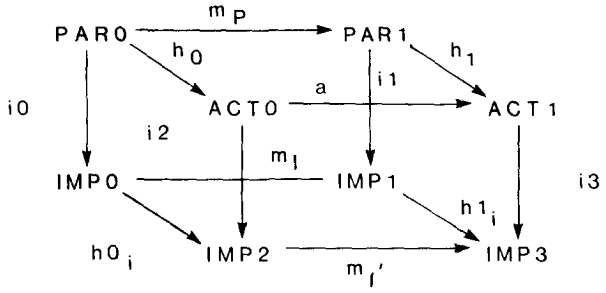
We now show that the operation of (standard) actualization is compatible with the notion of submodule.

3.6. THEOREM. *Let* MOD0 = (PAR0, IMP0, BOD0, EXP0) *be a submodule specification of* MOD1 = (PAR1, IMP1, BOD1, EXP1) *and* ACT0 *and* ACT1 *two actual parameter specifications with a specification morphism a*: ACT0 → ACT1. *Given parameter passing morphisms hj*: PARj → ACTj *for* $j = 0, 1$ *satisfying* $h1 \cdot m_P = a \cdot h0$ (consistency of parameter passing), MOD0(ACT0) *is a submodule specification of* MOD1(ACT1).

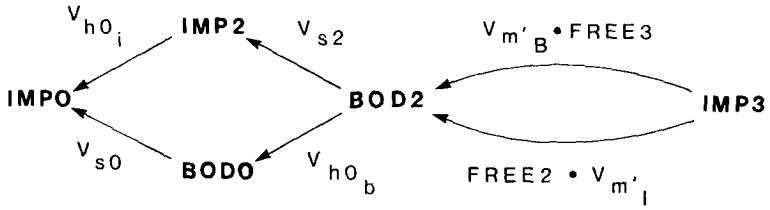
Remark. Recall that MODj(ACTj) is *not* the module specification act_{hj}(ACTj, MODj) resulting from "replacing" the parameter part with ACTj from which it differs in the first specification of the 4-tuple which is ACTj rather than $\varnothing$. Also the above situation includes the special case of ACT0 = ACT1, where $h0$ can be defined to be $h1 \cdot m_P$.

Proof of Theorem 3.6. For notational convenience, let MOD0(ACT0) = (ACT0, IMP2, BOD2, EXP2) and MOD1(ACT1) = (ACT1, IMP3, BOD3, EXP3) with

specification morphisms $i2$, $s2$, $v2$, $e2$ and $i3$, $s3$, $v3$, $e3$, respectively. Consider the diagram defining the new import interfaces



As in Definition 3.1, the back and top squares are commutative and the two side squares are pushout diagrams defining IMP2 and IMP3. Hence there exists a unique m'_I: IMP2 → IMP3 such that $m'_I \cdot h0_i = h1_i \cdot m_I$ and $m'_I \cdot i2 = i3 \cdot a$. Similarly, there exists a unique m'_E: EXP2 → EXP3 satisfying $m'_E \cdot e2 = e3 \cdot a$ and $m'_E \cdot h0_e = h1_e \cdot m_E$ and a unique m'_B: BOD2 → BOD3 with $m'_B \cdot h0_b = h1_b \cdot m_B$ and $m'_B \cdot s2 = s3 \cdot m'_I$. It is straightforward to check that the commutativity properties set forth in Definition 1.4 are satisfied. We only need to verify the semantical assumptions. First of all, since FREE0 and FREE1 are strongly persistent then, by the Extension Lemma in [9], so are FREE2 and FREE3. To show that $V_{m'_B} \cdot \text{FREE3} = \text{FREE2} \cdot V_{m'_I}$, consider the diagram



with BOD2 the pullback object. By using the strong persistency of the free functors, we have

$$V_{s2} \cdot V_{m'_B} \cdot \text{FREE3} = V_{s2} \cdot \text{FREE2} \cdot V_{m'_I},$$

$$V_{h0_b} \cdot V_{m'_B} \cdot \text{FREE3} = V_{h0_b} \cdot \text{FREE2} \cdot V_{m'_I},$$

and

$$V_{h0_i} \cdot V_{s2} \cdot V_{m'_B} \cdot \text{FREE3} = V_{s0} \cdot V_{h0_b} \cdot V_{m'_B} \cdot \text{FREE3}.$$

By the pullback property of **BOD2**, there can be only one such functor from **IMP3** to **BOD2**. Hence $V_{m'_B} \cdot \text{FREE3} = \text{FREE2} \cdot V_{m'_I}$. A similar argument shows that $V_{m'_E} \cdot R_{e1'} = R_{e0'} \cdot V_{m'_E}$.

3.7. COROLLARY. *If the specification* ACT1 *is an extension of the specification* ACT0 (*in particular of* ACT0 = ACT1) *then* $\text{act}_{h0}(\text{ACT0}, \text{MOD0})$ *is a submodule specification of* $\text{act}_{h1}(\text{ACT1}, \text{MOD1})$.

4. ACTUALIZATION AND UNION

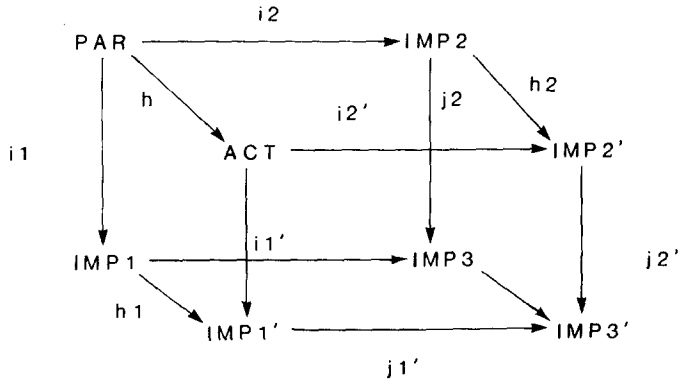
We now show that the operations of actualization and union of module specifications are compatible in the sense that an appropriate union of two modules actualized separately yields the same module that can be obtained by an appropriate actualization of their union. The first case we consider is that of two module specifications sharing a common parameter part.

4.1. THEOREM. *Let* MODi = (PAR, IMPi, BODi, EXPi) *for* $i = 1, 2$ *be module specifications and* h: PAR → ACT *a parameter passing morphism. Then*

$$\text{act}_h(\text{ACT}, \text{MOD1} +_{\text{PAR}} \text{MOD2}) = \text{act}_h(\text{ACT}, \text{MOD1}) +_{\text{ACT}\phi} \text{act}_h(\text{ACT}, \text{MOD2}),$$

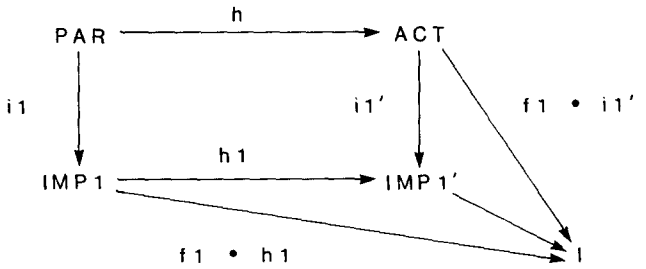
where ACT$\varnothing$ = ($\varnothing$, ACT, ACT, ACT).

Proof. Let MOD3 = MOD1 + $_{\text{PAR}}$ MOD2 = (PAR, IMP3, BOD3, EXP3) and MODi' = act_h(ACT, MODi) for $i = 1, 2, 3$. Let us prove first that the import interfaces of MOD3′ and MOD1′ + $_{\text{ACT}\phi}$ MOD2′ are the same. Consider the following diagram



The back, top, and left side squares are pushouts corresponding to the constructions of the import interfaces of MOD1 + $_{\text{PAR}}$ MOD2, act_h(ACT, MOD2) and act_h(ACT, MOD1), respectively. The "diagonal" square (PAR, ACT, IMP3, IMP3′) is the pushout diagram defining the import interface of act_h(ACT, MOD3). By the pushout properties of IMP1′ and IMP2′, there exist unique specification morphisms $j1'$: IMP1′ → IMP3′ and $j2'$: IMP2′ → IMP3′ such that the bottom and right side squares commute and the morphism $i3'$: ACT → IMP3′ *is* the diagonal of

the front square. Hence the front square is a commutative diagram. To show that $IMP3' = IMP1' +_{ACT} IMP2'$, let I be a specification and fi: $IMPi' \to I$ specification morphisms such that $f2 \cdot i2' = f1 \cdot i1'$. Then $f1 \cdot h1$: $IMP1 \to I$ and $f2 \cdot h2$: $IMP2 \to I$ satisfy $f2 \cdot h2 \cdot i2 = f1 \cdot h1 \cdot i1$ and therefore, since the back square is a pushout, there exists a unique $f3$: $IMP3 \to I$ such that $f3 \cdot ji = fi \cdot hi$, $i = 1, 2$. Since the diagonal diagram is a pushout, there is a unique $f3'$: $IMP3 \to I$ satisfying $f3 = f3' \cdot h3$ and $f3' \cdot i3' = f2 \cdot i2' = f1 \cdot i1'$. Finally, to show that $f3' \cdot j1' = f1$ it suffices to note that $f3' \cdot j1'$ and $f1$ make the two triangles in the diagram



commute. By uniqueness of the morphism from IMP1′ to I, the two morphisms must coincide. Similarly, $f3' \cdot j2' = f2$. Hence IMP3′ *is* the pushout object of IMP1′ and IMP2′ w.r.t. ACT. The proofs that $EXP3' = EXP1' +_{ACT} EXP2'$ and $BOD3' = BOD1' +_{ACT} BOD2'$ are the same by symmetry and by Lemma 3.2.

The next interesting case to consider is that of two module specifications $MODi = (PARi, IMPi, BODi, EXPi)$ sharing a common subparameter part PAR0 via specification morphisms pi: $PAR0 \to PARi$, $i = 1, 2$. Let ACT1 and ACT2 be actual parameter specifications with parameter passing morphisms hi: $PARi \to ACTi$. The union $MOD3 = MOD1 +_{PAR0} MOD2$ has $PAR3 = PAR1 +_{PAR0} PAR2$ as parameter part which must be actualized by a union of ACT1 and ACT2.

Each choice for this union leads to a different actualization of MOD3. Let ACT0 be a specification and let $h0$: $PAR0 \to ACT0$ and ki: $ACT0 \to ACTi$ be specification morphisms such that $hi \cdot pi = ki \cdot h0$, $i = 1, 2$. The morphism $h0$ represents the common actualization of PAR0 induced by the actualizations $h1$ and $h2$ through $p1$ and $p2$. The two extreme cases of the union $ACT1 +_{ACT0} ACT2$ correspond to choosing $ACT1 = ACT0 = ACT2$, where both PAR1 and PAR2 are intended to be actualized by the same specification, and to setting $ACT0 = PAR0$. In the latter case, the actualized union is not quite the "union" of the individual actualizations, due to the injectivity requirements in 1.4, but can be defined as a pushout in the category of module specifications and module morphisms [20].

4.2. THEOREM. *Let* $MOD3 = MOD1 +_{PAR0} MOD2$, $ACT3 = ACT1 +_{ACT0} ACT2$ *and* $h3 = h1 +_{h0} h2$ *be defined as above. Then*

$$act_{h3}(ACT3, MOD3) = act_{h1}(ACT1, MOD1) +_{ACT0\phi} act_{h2}(ACT2, MOD2).$$

The proof of this theorem is similar to the previous one and a special case of the next. An intuitive argument can be summarized as follows: the import interface IMP3′ of the left-hand side contains a copy of IMP3 = IMP1 + $_{\text{PAR0}}$IMP2 and a copy of ACT3 = ACT1 + $_{\text{ACT0}}$ACT2 with the PAR3 = PAR1 + $_{\text{PAR0}}$PAR2 part identified. Hence it contains a copy of IMP1 and ACT1 identified through PAR1, which is IMP1′ and, similarly, a copy of IMP2′. But a copy of IMP1 and one of IMP2 duplicate PAR0 while ACT1 and ACT2 have ACT0 in common. This duplication is eliminated by identifying the "gluing" of PAR0 and ACT0, which is just ACT0.

The most general situation is that of two module specifications MODi = (PARi, IMPi, BODi, EXPi), $i = 1, 2$, which share a common submodule MOD0 = (PAR0, IMP0, BOD0, EXP0) and which are actualized by hi: PAR$i \to$ ACTi, $i = 0, 1, 2$.

If the actualization is "uniform," that is, if ACT0 is a subspecification of ACT1 and ACT2, and $h0$ is the restriction of both $h1$ and $h2$ to ACT0, then the union of the individual actualizations is equal to the actualization of the union of the modules by the union of the actual parameters.

4.3. THEOREM. $\text{act}_{h1}(\text{ACT1}, \text{MOD1}) + _{\text{act}_{h0}(\text{ACT0}, \text{MOD0})}\text{act}_{h2}(\text{ACT2}, \text{MOD2}) = \text{act}_{h1 +_{h0} h2}(\text{ACT1} + _{\text{ACT0}}\text{ACT2}, \text{MOD1} + _{\text{MOD0}}\text{MOD2})$.

Proof. Let mi_I, mi_P, mi_B, and mi_E be the specification morphisms from MOD0 to MODi, $i = 1, 2$, as in Definition 1.4. Let fi: ACT0 $\to$ ACTi, $i = 1, 2$, be injective specification morphisms. The actualization is uniform if $fi \cdot h0 = hi \cdot mi_P$ for $i = 1, 2$.

We will restrict our attention to the export interfaces. The proofs for the import interfaces and for the body parts are identical by the symmetry of the roles of the interfaces and by Lemma 3.2, while no proof is necessary for the parameter parts, as they are all empty on both sides of the equation. Let, as before, act_{hi}(ACTi, MODi) = ($\varnothing$, IMPi', BODi', EXPi'), $i = 0, 1, 2$.

We need to show that EXP3′ = EXP3 + $_{\text{PAR3}}$ACT3 is equal to EXP1′ + $_{\text{EXP0}'}$EXP2′. To this extent, we will use a general result of category theory [18] on the commutativity of colimits. Let I and J be the following diagram categories

$$I = \text{ACT} \leftarrow \text{PAR} \to \text{EXP}, \qquad J = 1 \leftarrow 0 \to 2$$

and define a functor F: $I \times J \to$ CATSPEC on the objects by letting, for $i = 0, 1, 2$,

$$F(\text{ACT}, i) = \text{ACT}i$$
$$F(\text{PAR}, i) = \text{PAR}i$$
$$F(\text{EXP}, i) = \text{EXP}i$$

and on the morphisms by extending, using composition,

$$F(\mathrm{PAR} \to \mathrm{ACT}, \mathrm{id}_i) = hi$$

$$F(\mathrm{PAR} \to \mathrm{EXP}, \mathrm{id}_i) = ei$$

$$F(\mathrm{id}_{\mathrm{ACT}}, 0 \to i) = fi$$

$$F(\mathrm{id}_{\mathrm{PAR}}, 0 \to i) = mi_P$$

$$F(\mathrm{id}_{\mathrm{EXP}}, 0 \to i) = mi_E.$$

The extension is well defined since, for example, $F(\mathrm{PAR} \to \mathrm{EXP},\ 0 \to 1)$ can be obtained either as

$$F(\mathrm{id}_{\mathrm{EXP}}, 0 \to 1) \cdot F(\mathrm{PAR} \to \mathrm{EXP}, \mathrm{id}_0) = m1_E \cdot e0$$

or as

$$F(\mathrm{PAR} \to \mathrm{EXP}, \mathrm{id}_1) \cdot F(\mathrm{id}_{\mathrm{PAR}}, 0 \to 1) = e1 \cdot m0_P$$

and the two morphisms are the same by Definition 1.4. The assumption on the uniformity of the actualizations is used for $F(\mathrm{PAR} \to \mathrm{ACT},\ 0 \to i)$. Since CATSPEC is closed under pushouts, $\mathrm{Colim}_{I \times J} F$ exists and it is equal to both $\mathrm{Colim}_I(\mathrm{Colim}_J F)$ and $\mathrm{Colim}_J(\mathrm{Colim}_I F)$, that is, the colimit can be computed componentwise. But

$$\mathrm{Colim}_I(\mathrm{Colim}_J F) = \mathrm{Colim}_I(\mathrm{ACT3} \leftarrow \mathrm{PAR3} \to \mathrm{EXP3}) = \mathrm{EXP3}'$$

while

$$\mathrm{Colim}_J(\mathrm{Colim}_I F) = \mathrm{Colim}_J(\mathrm{EXP1}' \leftarrow \mathrm{EXP0}' \to \mathrm{EXP2}') = \mathrm{EXP1}' +_{\mathrm{EXP0}'} \mathrm{EXP2}'$$

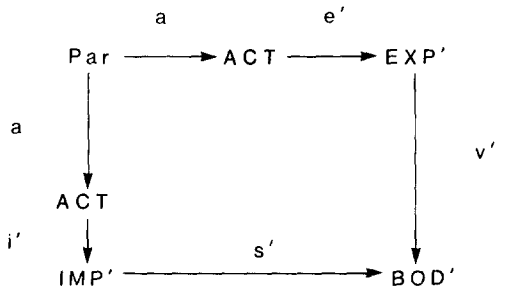
and the proof that the export interfaces are the same is complete. The specification morphisms $\mathrm{EXP3}' \to \mathrm{BOD3}'$ and $\mathrm{IMP3}' \to \mathrm{BOD3}'$ are also the same as they are uniquely induced by the universal property of EXP3′ and IMP3′, respectively.

In the previous theorems, we started with two module specifications MOD1 and MOD2 actualized separately and took their union and showed how to obtain an equivalent module specification by actualizing (by an appropriate actual parameter) the union of MOD1 and MOD2. For the opposite problem, i.e., decomposing the actualization of a union into a union of actualizations, let $\mathrm{MOD3} = \mathrm{MOD1} +_{\mathrm{MOD0}} \mathrm{MOD2}$ be actualized by $h3$: $\mathrm{PAR3} = \mathrm{PAR1} +_{\mathrm{PAR0}} \mathrm{PAR2} \to \mathrm{ACT}$. Since PAR3 is a p.o. object, $h3$ decomposes *uniquely* into $h1$: $\mathrm{PAR1} \to \mathrm{ACT}$ and $h2$: $\mathrm{PAR2} \to \mathrm{ACT}$, with common part $h0$: $\mathrm{PAR0} \to \mathrm{PAR1} \to \mathrm{ACT} = \mathrm{PAR0} \to \mathrm{PAR2} \to \mathrm{ACT}$. Then

$$\mathrm{act}_{h3}(\mathrm{ACT}, \mathrm{MOD3}) = \mathrm{act}_{h1}(\mathrm{ACT}, \mathrm{MOD1}) +_{\mathrm{act}_{h0}(\mathrm{ACT}, \mathrm{MOD0})} \mathrm{act}_{h2}(\mathrm{ACT}, \mathrm{MOD2}).$$

5. Parameterized Actualization

In the previous two sections, we only considered the simplest case of actualization by a single actual parameter. We now discuss a slightly more general case, where we allow actualization of a module specification by a parameterized specification, PS = (Par, ACT). Given a parameterized specification, i.e., a pair of specifications Par and ACT with an injective specification morphism a: Par → ACT (usually inclusion), the result of actualizing MOD = (PAR, IMP, BOD, EXP) by PS = (Par, ACT) w.r.t. a parameter passing morphism h: PAR → ACT is the module specification $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD}) = (\mathrm{Par}, \mathrm{IMP}', \mathrm{BOD}', \mathrm{EXP}')$ with the following commutative diagram



where IMP', BOD', and EXP' are as in Definition 3.1. Since the result in Proposition 3.3 does not depend on the morphism a, we still have (PAR, IMP, BOD, EXP) as a submodule specification of (ACT, IMP', BOD', EXP') and the result of Theorem 3.4 still holds.

5.1. Theorem. *The unrestricted semantics* SEM' *of* $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD})$ *is given by*

$$\mathrm{SEM}' = \mathrm{id}_{\mathrm{ACT}} +_{\mathrm{id}_{\mathrm{PAR}}} \mathrm{SEM}.$$

Unlike the situation with standard actualization, the actualized module specification has a parameter part Par. While the restriction functor R_e of MOD is compatible (Proposition 3.3) with the functor $R_{e'}$ of (ACT, IMP', BOD', EXP'), it need not be compatible with the restriction functor $R_{e' \cdot a}$ of (Par, IMP', BOD', EXP').

We can relate the restricted semantics of MOD and $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD})$ only under the additional condition of "parameter consistency."

5.2. Theorem. *Let* h: PAR → ACT *be a parameter passing morphism which is "parameter consistent," i.e., such that there is a specification morphism* p: PAR → Par *with* $a \cdot p = h$. *Then*

$$V_{he} \cdot \mathrm{RSEM}' = \mathrm{RSEM} \cdot V_{hi},$$

where RSEM *and* RSEM′ *are the restricted semantics of* MOD *and* $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD})$, *respectively.*

Proof. Since $V_{he} \cdot \mathrm{RSEM}' = V_{he} \cdot R_{e' \cdot a} \cdot \mathrm{SEM}'$, $\mathrm{RSEM} \cdot V_{hi} = R_e \cdot \mathrm{SEM} \cdot V_{hi}$ and by Theorem 5.1, $R_e \cdot \mathrm{SEM} \cdot V_{hi} = R_e \cdot V_{he} \cdot \mathrm{SEM}'$, it suffices to show that $V_{he} \cdot R_{e' \cdot a} = R_e \cdot V_{he}$. Let $A' \in \mathbf{EXP}'$:

(1) By definition of $R_{e' \cdot a}$, we have $V_{he}(R_{e' \cdot a}(A')) \subset V_{he}(A')$ and $V_e(V_{he}(R_{e' \cdot a}(A'))) = V_p(V_{e' \cdot a}(R_{e' \cdot a}(A'))) = V_p(V_{e' \cdot a}(A')) = V_e(V_{he}(A'))$. Hence

$$V_{he}(R_{e' \cdot a}(A')) \in \{B \in \mathbf{EXP}: B \subset V_{he}(A'),\ V_e(B) = V_e(V_{he}(A'))\}$$

and therefore

$$R_e(V_{he}(A')) \subset V_{he}(R_{e' \cdot a}(A')).$$

(2) By definition of R_e, we have $R_e(V_{he}(A')) \subset V_{he}(A')$ and by Proposition 3.3, $R_e(V_{he}(A')) = V_{he}(R_{e'}(A'))$. Notice now that $V_{e' \cdot a}(R_{e'}(A')) = V_a(V_{e'}(R_{e'}(A'))) = V_a(V_{e'}(A')) = V_{e' \cdot a}(A')$ and $R_{e'}(A') \subset A'$. Hence

$$R_{e'}(A') \in \{B \in \mathbf{EXP}': B \subset A',\ B_{\mathrm{Par}} = A'_{\mathrm{Par}}\}$$

and thus $R_{e' \cdot a}(A') \subset R_{e'}(A')$ which, in turn, implies $V_{he}(R_{e' \cdot a}(A')) \subset V_{he}(R_{e'}(A')) = R_e(V_{he}(A'))$.

The usefulness of the result in the last theorem is questionable (at best!) due to the additional assumption of parameter consistency, but it is the best possible if we limit the discussion to the basic algebraic case. The situation improves considerably if the interfaces can be described by algebraic specifications *and* other constraints [10, 6]. In particular, if the semantics of the parameterized specification PS is taken into account, then we can explicitly describe the restricted semantics RSEM′ of $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD})$ in terms of the restricted semantics RSEM of MOD. By considering parameterized specifications as special cases of module specifications with export = body and parameter = import, we follow [8, 9] in defining the semantics of PS = (Par, ACT) as the free (persistent) functor F_a: **Par** → **ACT**. After actualization, the semantics of the new import interface is no longer loose and the specification is augmented with constraints to allow only IMP′-algebras of the form $I +_P F_a(P')$, where $F_a(P')$ is the ACT-algebra freely generated by $P' \in \mathbf{Par}$. Under these conditions, it can be shown [6] that the restricted semantics of $\mathrm{act}_h(\mathrm{PS}, \mathrm{MOD})$ satisfies

$$\mathrm{RSEM}'(I +_P F_a(P')) = \mathrm{RSEM}(I) +_P F_a(P').$$

Note also that these conditions are compatible with the result in Theorem 3.5.

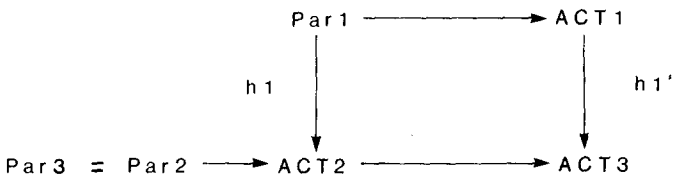
The interaction of parameterized actualization and the union of module specifications is easy to describe since all the results (and proofs) of Section 4 are still valid for the import, body, and export parts of the actualized module specifications, while the verification for the parameter part is straightforward. We will only state the most general result of the compatibility of parameterized actualization with the union of two modules sharing a common submodule.

Given parameterized specifications $PSj = (Parj, ACTj)$, $j = 0, 1, 2$, with aj: $Parj \to ACTj$, and specification morphisms pj: $Par0 \to Parj$ and qj: $ACT0 \to ACTj$ satisfying $aj \cdot pj = qj \cdot a0$, $j = 1, 2$, we can form the *union* $PS1 +_{PS0} PS2$ as in 2.4.

5.3. THEOREM. *Let* MOD0 *be a submodule specification of* MOD1 *and* MOD2, PS0 *a parameterized subspecification of* PS1 *and* PS2 *and let hj*: $PARj \to ACTj$, $j = 0, 1, 2$, *be uniform parameter passing morphisms. Then*

$$act_{h1}(PS1, MOD1) +_{act_{h0}(PS0, MOD0)} act_{h2}(PS2, MOD2)$$
$$= act_{h1 +_{h0} h2}(PS1 +_{PS0} PS2, MOD1 +_{MOD0} MOD2).$$

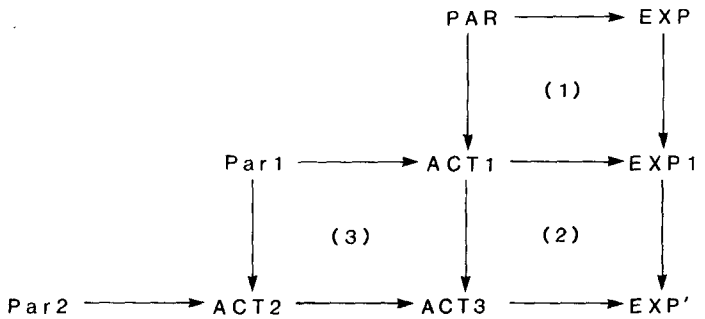
Unlike the standard case discussed in Section 3, a parameterized actualization of MOD provides a module with a new nonempty parameter part which can be actualized again by another parameter passing morphism. As the last result of this paper, we will show that successive actualizations are equivalent to a single actualization by an appropriate actual parameter. First, we review the parameter passing mechanism in [8, 9] for parameterized specifications. Given $PS1 = (Par1, ACT1)$, $PS2 = (Par2, ACT2)$ and a specification morphism $h1: Par1 \to ACT2$, the result of passing PS2 to PS1 by $h1$, denoted by $PS1 *_{h1} PS2$, is the parameterized specification $PS3 = (Par3, ACT3)$ as in the following diagram



5.4. THEOREM. *Let* MOD *be a module specification,* $PSj = (Parj, ACTj)$, $j = 1, 2$, *parameterized specifications, and* $h: PAR \to ACT1$ *and* $h1: Par1 \to ACT2$ *specification morphisms. Then*

$$act_{h1}(PS2, act_h(PS1, MOD)) = act_{h1' \cdot h}(PS1 *_{h1} PS2, MOD).$$

Proof. Once more, we will discuss only the export interfaces. Let $MOD1 = act_h(PS1, MOD)$ and consider the following diagram



If (1) and (2)+(3) are pushouts, then EXP′ is the export interface of act_{h1}(PS2, act_h(PS1, MOD)) by definition of actualization. But (3) is a pushout by definition of PS1 $*_{h1}$ PS2 and therefore so is (2) by standard properties of pushouts. Since composition of pushouts is a pushout, so is (1)+(2), which is exactly the definition of the export interface of $act_{h1'.h}$(PS1 $*_{h1}$ PS2, MOD).

We close this paper with a few remarks. First of all we have considered here the operation of actualization only between plain specifications (parameterized and not) and modules. From a methodological point of view, there is no reason why the actualization of a module MOD by (the export interface of) another module MOD′ should not be included, and, in fact, [11] contains a proposal for such an operation, in the special case where the import interface and parameter part of MOD′ coincide. The general case cannot be handled with basic algebraic specifications but requires the use of additional generating or logical constraints [6, 10]. This type of extension is being investigated.

The interaction between union and actualization presented here is only part of the work done in establishing the compatibility properties of the basic interconnections of module specifications [5, 20]. The compatibility of composition with standard actualization was already shown in [11]. Compatibilities are essential to guarantee that the modular structuring of large software systems does not depend on the order in which the operations are applied.

We have used the analogy of Ada packages in Section 1 to illustrate the module concept: a more detailed comparison can be found in [1], while [12] discusses the similarities with Modula-2 modules. Among other proposals of module definitions, we should mention OBJ2 [13] and Extended ML [23], in which the notion of a module includes semantical conditions on the interfaces, not present in Ada and Modula-2.

ACKNOWLEDGMENTS

I am indebted to H. Ehrig and E. K. Blum for many helpful discussions. I am also grateful to the anonymous referee for the useful comments and constructive criticism on the earlier version of this paper. The research was partially supported by the National Science Foundation under Grants MCS-8203666 and DCR-8406920.

REFERENCES

1. E. K. BLUM, H. EHRIG, AND F. PARISI-PRESICCE, Algebraic specification of modules and their basic interconnections, *J. Comput. System Sci.*, **34**, 293–339.
2. E. K. BLUM AND F. PARISI-PRESICCE, The semantics of shared submodule specifications, *in* "Proceedings, CAAP '85," Lect. Notes in Comput. Sci. Vol. 185, pp. 359–373, Springer-Verlag, New York/Berlin, 1985.
3. R. M. BURSTALL AND J. A. GOGUEN, Putting theories together to make specifications, *in* "Proceedings, 5th Internat. Joint Conf. on Artif. Intell., Cambridge 1977," pp. 1045–1058.

4. H.-D. EHRICH, On the theory of specification, implementation and parameterization of abstract data types, *J. Assoc. Comput. Mach.* **29**, No. 1 (1982), 206–227.
5. H. EHRIG, W. FEY, AND F. PARISI-PRESICCE, Distributive laws for composition and union of module specifications for software systems, *in* "Proceedings, IFIP TC2 Work. Conf. on Program Specification and Transformation, Bad Tolz, April 1986."
6. H. EHRIG, W. FEY, F. PARISI-PRESICCE, AND E. K. BLUM, Algebraic Theory of Module Specifications with Constraints, *in* "Proceedings, Math. Found. of Comput. Sci., Lect. Notes in Comput. Sci., Vol. 233, pp. 59–77, Springer-Verlag, New York/Berlin, 1986.
7. H. EHRIG AND H.-J. KREOWSKI, Compatibility of parameter passing and implementation of parametrized data types, *Theoret. Comput. Sci.* **27** (1983), 255–286.
8. H. EHRIG, H.-J. KREOWSKI, J. W. THATCHER, E. G. WAGNER AND J. B. WRIGHT, Parameter passing in algebraic specification languages, *in* "Proceedings, Aarhus Workshop on Prog. Spec., 1981," Lect. Notes in Comput. Sci. Vol. 134, pp. 322–369, Springer-Verlag, New York/Berlin, 1982.
9. H. EHRIG AND B. MAHR, "Foundations of Algebraic Specifications 1: Equations and Initial Semantics," EATCS Monographs on Theoret. Comput. Sci. Vol. 6, Springer-Verlag, Berlin/New York, 1985.
10. H. EHRIG, E. G. WAGNER AND J. W. THATCHER, Algebraic specifications with generating constraints, *in* "ICALP 83," Lect. Notes in Comput. Sci. Vol. 154, pp. 118–202, Springer-Verlag, New York/Berlin, 1983.
11. H. EHRIG AND H. WEBER, Algebraic specification of modules, *in* "Proceedings, IFIP Working Conference on Formal Models in Programming," (E. J. Neuhold and G. Chronist, Eds.), pp. 231–258, North-Holland, Amsterdam, 1985.
12. H. EHRIG AND H. WEBER, Programming in the large with algebraic module specifications, *in* "Proceedings, IFIP Congress '86, Dublin, September 1986."
13. K. FUTATSUGI, J. A. GOGUEN, J.-P. JOUANNAUD, AND J. MESEGUER, Principles of OBJ2, *in* "12th ACM POPL, New Orleans, 1985," pp. 52–66.
14. H. GANZINGER, Parameterized specifications: Parameter passing and implementation, *ACM TOPLAS* **5**, No. 3 (1983).
15. V. GIARRATANA, F. GIMONA, AND U. MONTANARI, Observability concepts in abstract data type specifications, *in* "5th Symp. Math. Found. of Comp. Sci. 1976," Lect. Notes in Comput. Sci. Vol. 45, pp. 576–587, Springer-Verlag, New York/Berlin, 1976.
16. J. A. GOGUEN AND J. MESEGUER, Universal realization, persistent interconnection and implementation of abstract modules, *in* "ICALP 82," Lect. Notes in Comput. Sci. Vol. 140, pp. 265–281, Springer-Verlag, New York/Berlin, 1982.
17. J. A. GOGUEN, J. W. THATCHER, AND E. G. WAGNER, An initial algebra approach to the specification, correctness and implementation of abstract data types, *in* "Current Trends in Prog. Method., IV: Data Structuring" (R. T. Yeh, Ed.), pp. 80–149, Prentice-Hall, Englewood Cliffs, N.J., (1978).
18. H. HERRLICH AND G. E. STRECKER, "Category Theory," Allyn & Bacon, Boston, 1973.
19. B. H. LISKOV AND S. N. ZILLES, Specification techniques for data abstraction, *IEEE Trans. Software Engrg.* **SE-1**, No. 1 (1975), 7–19.
20. F. PARISI-PRESICCE, Inner and mutual compatibility of basic operations on module specifications, *in* "Proceedings, CAAP 86," Lect. Notes in Comput. Sci. Vol. 214, pp. 30–44, Springer-Verlag, New York/Berlin, 1986; full version: Tech. Rep. 86-06, Tech. Univ. Berlin, April 1986.
21. D. L. PARNAS, A technique for software module specification with examples, *Comm. ACM* **15**, No. 5 (1972), 330–336.
22. H. REICHEL, Behavioral equivalence—A unifying concept for initial and final specification methods, *in* "Proceedings, 3rd Hungarian Comput. Sci. Conf., Budapest, 1981," pp. 27–39.
23. D. SANNELLA AND A. TARLECKI, Program specification and development in standard ML, *in* "12th ACM POPL, New Orleans, 1985," pp. 67–77.
24. D. SANNELLA AND A. TARLECKI, On observational equivalence and algebraic specification, *in* "CAAP 85," Lect. Notes in Comput. Sci. Vol. 185, pp. 308–322, Springer-Verlag, New York/Berlin, 1985.

25. D. SANNELLA AND M. WIRSING, "A Kernel Language for Algebraic Specification and Implementation," Internal Report CSR-131-83, pp. 1–44, Univ. Edinburgh, 1983.
26. "The Programming Language Ada," reference manual, Lect. Notes in Comput. Sci. Vol. 106, Springer-Verlag, New York/Berlin, 1981.
27. H. WEBER AND H. EHRIG, Specification of Modular Systems, *IEEE Trans. Software Engrg.*, June 1986.